ANNUAL PROGRESS REPORT

RESEARCH
ON
AUTOMATIC CLASSIFICATION,
INDEXING AND EXTRACTING

F. T. Baker
J. H. Williams, Jr.

CONTRACT NONR 4456(00)

Submitted to:

Information Systems Branch
Office of Naval Research
Department of the Navy
Washington, D. C. 20360

Prepared by:

Federal Systems Division
International Business Machines Corporation
Gaithersburg, Maryland 20760

August 1968

52

ABSTRACT

To support studies in automatic indexing, classification and extracting, a general purpose frequency program has been developed to further our theoretical and practical understanding of text word distributions. While the program is primarily designed for counting strings of character-oriented data, it can be used without change for counting any items which can be represented in an integral number of characters. Counts may be obtained simultaneously at several levels of detail, such as for sentences, paragraphs, chapters and entire documents. Both printed outputs and outputs for further computer processing may be obtained, and a variety of summary and detailed outputs are available.

The program, titled FRQNCY, is written in the Fortran IV language and has been compiled and run on the IBM System/360 using Fortran IV (G) and the System/360 Operating System. It uses at least one feature of the IBM System/360 Fortran IV language (LOGICAL*1 and INTEGER*2 variables) which is not in USASI Fortran and hence may not compile or run under other Fortran systems. The program is extensively parametrized to allow its efficient use on computers with varying amounts of immediate-access storage and input/output equipment.

This report is a complete writeup of the frequency program. It covers the purpose and usage of the program and also describes its organization and internal operation. Finally, guidelines for modifying the program or adapting it to different computers are also included.

PROGRESS SUMMARY

This year's effort consisted of writing and testing the general purpose frequency program whose functions, organization and specifications are described in a previous report: "Research on Automatic Classification, Indexing, and Extracting" by F. T. Baker, G. L. Johnson, M. Jones, and J. H. Williams, AD 485, 188.

Attached to this report is a complete writeup on the frequency program and its use. The purpose of the program is reviewed in Section 1. The inputs and outputs are described in Sections 2 and 3. Operating instructions including control card formats are shown in Section 4. The restart procedure is described in Section 5. Requirements for the input item identification subroutine are given in Section 6. Section 7 covers the organization of the program, and Section 8 describes ways to modify the program or adapt it to different computers. Restrictions are discussed in Section 9. Section 10 contains an example describing a typical application and the setup necessary to run it.

FRQNCY

A General-Purpose Frequency Program

Written under:

Contract NONR 4456(00)
Information Systems Branch
Office of Naval Research
Department of the Navy
Washington, D. C. 20360

Federal Systems Division
International Business Machines Corporation
Gaithersburg, Maryland 20760

August 1968

CONTENTS

1. INTRODUCTION

FRQNCY is a general-purpose frequency program for producing lists of items and counts of the number of times they occur. While it is primarily designed for counting strings of character-oriented data (e.g., words in text, syllables in a dictionary), the program can be used without change for counting any type of items, each of which contains an integral number of bytes. If the original data is hierarchically structured, then counts may be made by unit at any or all levels* of the structure. For example, a document (first-level unit) may be considered to consist of chapters (second-level units), each of which has a number of paragraphs (third-level units), consisting of sentences (fourth-level units), containing individual words. If words are the items to be counted, then one may obtain word counts for each sentence, paragraph and chapter, as well as for the document as a whole.

Either a sequential data set for listing or a sequential data set for further computer processing (or both) may be obtained as detailed output for each unit of counting at any level of count. The item

*Note: The words "level" and "interval" should be considered synonymous in this document and in the FRQNCY program itself.

itself may be accompanied by any or all of the following:

1. Its length in bytes.

2. Its absolute frequency (e.g., ten times in this chapter).

3. Its relative frequency (e.g., 2.3% of the words in this chapter).

4. The position of its first occurrence (e.g., 103rd word of the chapter).

5. The number of different immediately-lower-level units in which it occurred (e.g., in four paragraphs in this chapter).

Similarly, eight types of summary outputs are available for each unit of counting at any level of count. They are:

1. Total number of items ("tokens").

2. Number of unique items ("types").

3. Distribution of the number of unique items occurring once, twice, thrice, etc., ("type/token distribution").

4. Distribution of tokens by the first byte (up to eight bits) of the item ("type/first character distribution").

5. Distribution of types by the first byte (up to eight bits) of the item ("token/first character distribution").

6. Distribution of tokens by the length of the item ("token/item length distribution").

7. Distribution of types by the length of the item ("type/item length distribution").

8. Distribution of not-previously-occurring types ("new types") per fixed number of tokens ("growth rate distribution").

FRQNCY is written in the Fortran IV language and has been compiled and run on the IBM System/360 Fortran IV (G) and the System/360 Operating System. It uses at least one feature of the IBM System/360 Fortran IV language (LOGICAL*1 and INTEGER*2 variables) which is not in USASI Fortran and hence may not compile or run under other Fortran systems. The program is extensively parametrized to allow its efficient use on computers with varying a.nounts of immediate-access storage and input/output equipment.

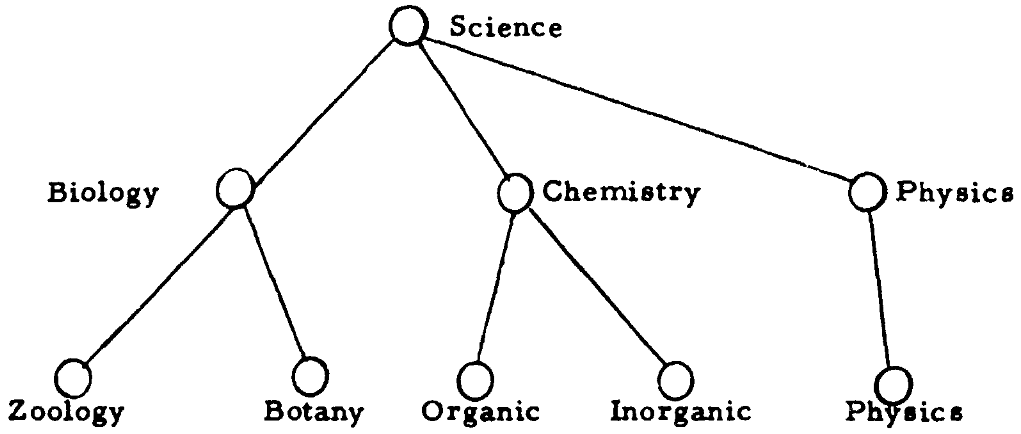
The FRQNCY program was produced by the IBM Federal Systems Division under Contract NONR 4456(00) of the Office of Naval Research. It is not a part of the IBM product line and has not been subjected to any formal product tests. Potential users should make any necessary evaluations concerning the utility of FRQNCY within their environments. While there is no maintenance provided or planned for FRQNCY, users are welcome to discuss any problems they may encounter with the author:

F. T. Baker
IBM Corporation
Federal Systems Division
18100 Frederick Pike
Gaithersburg, Maryland 20760

2. INPUT

Because of the great variety of types of input data from which users might desire items to be counted, it was impracticable to try to specify a single input processing algorithm. Instead, FRQNCY assumes that a subroutine will be provided which, whenever called, either passes to FRQNCY a single input item or notifies FRQNCY that a unit of counting (or the entire input) is completed. In this way the user is totally unrestricted as to the type of item counted or the rules for isolating it. As a result of this approach, FRQNCY itself never refers to the original input data directly but instead uses the items transmitted to it by the subroutine. The subroutine, which must be named SETITM, is responsible for all manipulation of any input data set(s) used. Complete instructions for writing a SETITM subroutine and interfacing it with FRQNCY are presented in Section 6.

The only restrictions on items to be counted are that they must be strings of bytes less than 256 bytes in length and that input units must be presented in hierarchical sequence. For an illustration of the latter requirement, suppose that words are to be counted by document and by category within a hierarchical structure such as the following:



and that fourteen input documents belong to the categories as follcws:

Document Number	Third-Level Category	Second-Level Category	First-Level Category
1	Inorganic	Chemistry	Science
2	Botany	Biology	Science
3	Organic	Chemistry	Science
4	Zoology	Biology	Science
5	Inorganic	Chemistry	Science
6	Inorganic	Chemistry	Science
7	Zoology	Biology	Science
8	Physics	Physics	Science
9	Organic	Chemistry	Science
10	Botany	Biology	Science
11	Physics	Physics	Science
12	Organic	Chemistry	Science
13	Zoology	Biology	Science
14	Inorganic	Chemistry	Science

Then all words from the first Zoology document (4) must be presented, followed by all words for the second Zoology document (7) followed by all words for the third Zoology document (13). All words for each of the Botany documents (2 and 10) must be presented next, followed by all words for each of the Organic documents (3, 9, and 12), followed by all words for each of the Inorganic documents (1, 5, 6, and 14). Finally, all words for each of the Physics documents (8, 11) are presented.

The reason for the two Physics categories in the above structure is the necessity to present all documents to FRQNCY at the lowest level. Thus, if the tree structure desired has some leaves not at the lowest level, it must be extended through use of repetitive or "dummy" nodes. The only structures allowable are therefore trees with all leaves at the same (lowest) level.

3. OUTPUTS

As mentioned in Section 1, two types of detailed output are available at each level of the structure, a data set for listing or a data set for further processing. The format of the printed listing is self-explanatory. At the top of each page appears a line containing the identification field for each level down to and including the level of the unit whose items are being counted. Next follows a header line containing an identification of each column of data. Below this is a line for each input item in that unit, containing the desired counts at the left and the item itself at the right. Items appear in collating sequence within the listing for each unit, and a new page is begun for each unit.

The sequential data set which may be produced at each level for further processing contains information for many units of counting, each of which contains many items. Therefore, for each unit, a header record in one format is followed by detail records and a trailer record in a second format. The header identifies the unit, the detail records contain the requested data about items in that unit, and the trailer record warns that another header (or the end of the data set) follows.

The format of the header is as follows:

Field	Length in Bytes
Total Frequency	4
Unit Identification	NID*NINTVS (see below)

The Total Frequency field contains an INTEGER*4 variable which represents the total frequency of all items in the following unit. The Unit Identification field consists of a string of bytes which identify the following unit. This string consists of NINTVS substrings, each NID bytes long, where NINTVS is the number of levels in the structure and NID the maximum number of bytes in the identification for any level. (See Section 5 for additional information on NID and NINTVS.) The order of the substrings is from the major level of the structure to the minor (level 1 to level NINTVS).

The format of the detail records comprising the data about items within the unit is as follows:

Field	Length in Bytes
Item Length (if specified)	2
Raw Frequency (if specified, or if Relative Frequency not specified)	4
Relative Frequency (if specified)	4
First Occurrence (if specified)	4
Lower Level Uses (if specified)	4
Item (always present)	Variable

The Item Length field (optional) contains an INTEGER*2 variable which gives the number of bytes in the item itself. The Raw Frequency field (always present unless Relative Frequency is specified, in which case it is optional) contains an INTEGER*4 variable representing the number of times the item occurred in this unit. The Relative Frequency field (optional) contains a REAL*4 variable representing the Raw Frequency divided by the Total Frequency. The First Occurrence field (optional) contains an INTEGER*4 variable representing the position (in terms of number of items) within this unit at which this item first occurred. The Lower Level Uses field (optional) contains an INTEGER*4 variable representing the number of immediately-lower-level units in which this item occurred. The Item field contains the item itself and is as many bytes long as initially specified when the item was passed to FRQNCY by SETITM (see Section 6). Records for items within each unit appear in item collating sequence.

The trailer record is the same format as the item record; in other words, it contains the same fields with the same length and in the same order as the data item format selected for the current level. It can be distinguished from a data item in that the Raw Frequency field (or Relative Frequency field if Raw Frequency was not chosen) contains zero, which cannot occur for a legal item. The Length field (if specified) contains one, and the Item field contains an all-zero byte. Any other fields specified contain zeroes. This record was designed so that it could be read using the same READ and FORMAT statements as the data items and then distinguished by a test for a zero Frequency field. (This procedure is generally necessary since the number of items in a unit is usually variable.)

The second major type of output which is obtainable from FRQNCY consists of the eight forms of summary outputs described in Section 1. A single data set for listing is used to contain all the summary outputs selected for a given level of the structure. For each unit, those outputs selected occur in the order in which they are described below. A new page is begun for each type of output selected, and on each page appears at the top a line containing the identification field for each level down to and including the level of the current unit, as well as a line identifying the type itself. Descriptions of the formats for each type of output follow:

1. The total number of tokens is printed as a single line.

2. The total number of types is printed as a single line.

3. The type/token distribution is printed in a tabular arrangement. The first line contains the number of types occurring once, twice,....nine times; the second contains the number occurring ten, eleven,....nineteen times; and so on. The number of occurrences for the first entry within a row is printed at the left of the row, and the columns are headed by digits zero, one,....nine. Any line with no occurrences is omitted. The table currently is set up to contain

types occurring up to 999 times; the number of types occurring more than this is placed in the "zero" position of the table.

4. The type/first character distribution is printed in a tabular arrangement. Each entry represents the number of types in which a particular bit configuration occurred as the first byte. The first five bits of the byte (or as many as are present) are printed at the left of the rows, and the last three bits head the columns. Rows with no entries are not printed.

5. The token/first character distribution is printed in the same format as the type/first character distribution. In this table, each entry represents the number of tokens in which a particular bit configuration occurred as the first byte.

6. The type/item length distribution is printed in a tabular arrangement. The first line contains the number of types with byte length one, two,....nine (zero cannot occur); the second contains the number of types with byte lengths ten, eleven,....nineteen; and so on. The byte length of the first entry within a row is printed at the left of the row, and the columns are headed by the digits zero, one,....nine. Rows with no entries are not printed.

7. The token/item length distribution is printed in the same format as the type/item length distribution. In this table, each entry represents the number of tokens with a particular byte length.

8. The growth rate distribution is printed as a vertically-organized array. Each line contains information about the number of new types (types not previously encountered) occurring within a fixed interval of tokens specified as an input parameter. For example, if the number of tokens specified was 500, then the first line contains information about types which occurred in the first 500 tokens; the second line contains information about types which did not occur in the first 500 tokens but did occur in the second 500 tokens; and so on. There are five entries on each line. The first identifies the number of the last token in the current interval (e.g., 500, 1000, 1500, etc.). The second identifies the total number of types occurring up to that point, and the third identifies the number of new types occurring in the current interval. The fourth line identifies the ratio of the number of new types in

this interval to the total number of types, and the fifth line represents the difference between the number of new types occurring in the last interval and the number occurring in this interval.

4. OPERATION

The basic form of the FRQNCY program is a FORTRAN source program consisting of a main program and a set of subroutines. To this program must be added the user's input item identification subroutine SETITM (see Section 6 for details). The program may then be compiled and run.

When running FRQNCY, the user must supply a number of data sets. With the exception of the Parameter data set, the location and numbering of these data sets is completely up to the user. They are all sequential data sets and may be placed on tape, disk, etc.; the output data sets may also be written directly to a printer if desired. The Parameter data set is always assumed to be data set number 5; it is the first data set read by FRQNCY and contains information on the particular run options and data set numbering desired by the user.

FRQNCY is organized much like a typical sort-merge program. Initially, a set of parameters are read and processed to specialize the program to the run desired. Next, the user's SETITM subroutine is called repetitively to supply FRQNCY with input items; these are sequenced and counted until a lowest-level input unit is complete or until the available fast-access storage is full. At these points the sequenced strings of items and their counts are placed on sequential data sets for future merging. When the input is completed (signaled by SETITM) the merge is begun. First, lowest-level units are completely sequenced and any desired outputs written. The process is then repeated for the next-higher-level units until all units at the highest level with any output requested have been sequenced. This makes it possible to count individual units only, if desired, since "dummy" level(s) with no outputs specified can be provided to complete the tree structure required by the program.

The Parameter data set, as mentioned above, is always data set 5 and contains all the setup information for FRQNCY. It must consist of 80-byte card format records and must contain one General Option Card, and a Level Option Card for each level of input data structure (including any "dummy" levels). It may optionally contain Initial Dictionary Item Cards if desired. Each of these cards will be discussed separately.

The first card of the Parameter data set must be the General Option Card. It has the following format:

Card Cols.	Information
1-5	The number of levels in the input structure; also the number of Level Option Cards.
6-10	The number of Initial Dictionary Item Cards; if none are supplied, this should be zero (Ø).

Card Cols. (Contd.)	Information
11-15	The maximum length of an internal chain of sequenced items; using FRQNCY as supplied, eight (8) is an appropriate value for this field.
16-20	The order of the merge; twice this many Merge data sets must be supplied. The minimum is two (2) and the maximum is four (4) in FRQNCY as supplied.
21-25	The number of the first Merge data set. Numbering of Merge data sets is assumed to be sequential, beginning with this number.
26-30	Number of lines per printed page; any output data set for printing will be formatted (by use of carriage control characters) not to exceed this value. The minimum value is four (4) plus the number of levels; there is no maximum.
31-35	Maximum length of any level identification field. The user's SETITM subroutine must provide an identification byte string for each input unit; this is the longest one to be provided. With FRQNCY as supplied, the minimum value is one (1) and the maximum is sixteen (16).
36-40	The number of the Comments data set. This is a sequential data set in which information on the progress of the program is placed; hence, an on-line printer or typewriter is a logical choice. This data set contains carriage control characters for formatting purposes.
41-45	Reserved for future use; must be set to zero (Ø).
47	Reserved for future use; must be set to the letter F.
49	Reserved for future use; must be set to the letter F.

Following the General Option Card must be a series of Level Option Cards. There must be one card for each level of structure, and they must appear in sequence from the highest level (the root of the tree; level number (1)), to the lowest (the leaves of the tree, the highest level number). The Level Option Cards contain the name of each level and the output options for them. Their format is as follows:

Card Cols.	Information
1-16	Name of the level; this is placed on any output for printing together with the SETITM-supplied identification of the individual units at that level. For example, DOCUMENT might be the name of level one (1), CHAPTER of level two (2), PARAGRAPH of level three (3) and SENTENCE of level four (4). SETITM would then supply an identification field for the document, chapter, paragraph and sentence being counted.
17-19	Number of the data set to be used for summary outputs at this level. This is a formatted, 132 character-per-line, sequential data set with carriage control characters, designed to be printed. If no summary outputs are desired at this level, this field should be zero (Ø), and Columns 2Ø-36 need not contain data.
20	Must contain the letter T if total types are desired for units at this level; the letter F if not.
21	Must contain the letter T if total tokens are desired for units at this level; the letter F if not.

Card Cols. (Contd.)	Information
22	Must contain the letter T if type/token distributions are desired for units at this level; the letter F if not.
23	Must contain the letter T if type/first character distributions are desired for units at this level; the letter F if not.
24	Must contain the letter T if token/first character distributions are desired for units at this level; the letter F if not.
25	Must contain the letter T if type/item length distributions are desired for units at this level; the letter F if not.
26	Must contain the letter T if token/item length distributions are desired for units at this level; the letter F if not.
27-36	Must contain the token interval over which new types are to be counted if growth rate distributions are desired for units at this level; otherwise must contain zero (Ø).
37-39	Number of the data set to be used for detailed print outputs for units at this level. This is a formatted, 132 character-per-line, sequential data set witb carriage control characters. If no detailed print outputs are desired at this level, this field should be zero (Ø), and Columns 4Ø-44 need not contain data.
40	Must contain the letter T if item lengths are to be printed; the letter F if not.
41	Must contain the letter T if raw frequencies are to be printed; the letter F if not.

Card Cols. (Contd.)	Information
42	Must contain the letter T if relative frequencies are to be printed; the letter F if not. (If both Columns 41 and 42 contain F, T is assumed for Column 41.)
43	Must contain the letter T if the locations of item's first uses in units at this level are to be printed; the letter F if not.
44	Must contain the letter T if the number of next-lower-level units in which this item occurred are to be printed; the letter F if not. (F is always assumed for the lowest level.)
45-47	Number of the data set to be used for detail output for further processing for units at this level. This sequential data set will contain variable-length records whose formats are described in Section 3. If no output for further processing is desired from this level, this field should be zero (Ø), and Columns 48-52 need not contain data.
48	Must contain the letter T if item lengths are to be provided for further processing; the letter F if not.
49	Must contain the letter T if raw frequencies are to be provided for further processing; the letter F if not.
50	Must contain the letter T if relative frequencies are to be provided for further processing; the letter F if not. (If both Columns 49 and 50 contain F, T is assumed for Column 49.)
51	Must contain the letter T if the location of items' first uses in units at this level are to be provided for further processing; the letter F if not.

Card Cols. (Contd.)	Information
52	Must contain the letter T if the number of next-lower-level units in which this item occurred are to be provided for further processing; the letter F if not. (F is always assumed for the lowest level.)

Following the Level Option Cards, a series of Initial Dictionary Item Cards may be provided. Although not required, the inclusion of high-frequency items on these cards will speed processing of the input. Thus, if such items can be identified in advance of a run, they may be specified on these cards. In FRQNCY as supplied, up to 999 of these items may be specified. They must be ordered by collating sequence of the items, and the number of items must be provided in Columns 6-10 of the General Option Card. The format of the Initial Dictionary Item Cards is as follows:

Card Cols.	Information
1-3	Length of the item in bytes.
4-80	The item itself. If the item is more than 77 bytes long, it may be continued in Columns 1-80 of succeeding cards up to a maximum of 255 bytes.

Once the above cards have been read and checked, FRQNCY begins to process input items. No other control information is required by FRQNCY itself, although additional information describing data sets, compilation options, etc., may be required by an operating system. One point which should be noted is that FRQNCY is designed to support additional options in later versions. As a result, some unresolved symbol references exist which will eventually be subroutine names. Although these subroutines are not called by the present version, it may be necessary to notify an operating system that these unresolved references may be ignored. In the version of FRQNCY provided, the following six symbols represent such unresolved references:

INSPIT
ININEX
INCONC
INEXM
SPITM
CONCOR

Section 10 contains an example of FRQNCY setup and the OS/360 setup necessary for a typical run.

5. RESTART

Because FRQNCY may run for a considerable length of time if the input is voluminous or if many levels are involved, a restart feature has been provided. Using this feature, the program may be restarted at the beginning of any merge pass (including the first one, immediately following the sort phase). At the beginning of each merge pass, FRQNCY places in the Comments data set a message giving all the data needed to restart at that point. This information may be added to the General Option Card to perform a restart in place of a normal run.

More specifically, at the beginning of each merge pass, FRQNCY provides for potential restart purposes, the level number, the number of the first Merge data set to that merge pass, the largest number of strings remaining to be merged at that level, and the length of the longest item. If the run aborts or is stopped during that pass, it may be restarted using those four parameters and the Merge data sets which were input to that pass. Only the last such set of data may be used, since the logic of the merge is such that the output from each merge pass is written over the input to the previous pass.

To perform a restart, the setup for the run must be identical to the original run with two exceptions. The first exception is the addition of four fields of restart data to the General Option Card as follows:

Card Cols.	Information
50-54	Level number at which restart is to begin.
55-59	First Merge data set to this pass.
60-64	Largest number of strings remaining to be merged at this level.
65-69	Length of the longest input item.

The second exception is that Input Dictionary Item Cards (if any) need not be provided in the rerun. With this setup, FRQNCY will space data sets up to the points they were at when that merge pass began, and then continue the run. No input data set need be mounted for a rerun, since FRQNCY can be restarted only after input processing is complete.

6. SETITM SUBROUTINE REQUIREMENTS

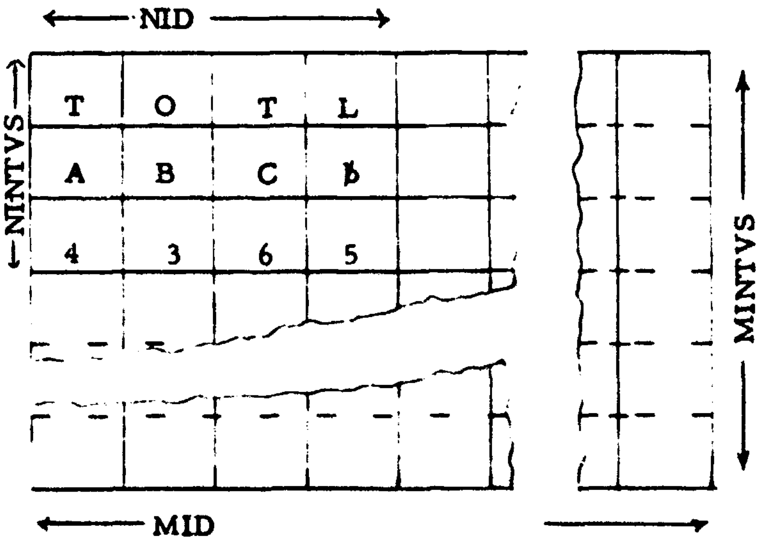
The SETITM subroutine is written by each user to scan his own input. In this subroutine the user defines his own item identification rules and exceptions. The output of the subroutine to the main FRQNCY program is the actual items to be counted.

There are two entries to SETITM, an initialization entry and a normal entry. The purpose of the initialization entry is to make array and item parameters known within SETITM and to allow the user to prepare his data set for processing. The purpose of the normal entry is to allow identification of items for FRQNCY processing, of unit breaks and of the end of the input.

The initialization entry should have the following format:

```
SUBROUTINE SETITM (ITEM, /MLNGTH/, /LNGTH/, NEWID,
                   /MID/, /NID/, /MINTVS/, /NINTVS/,
                   /ICNT/)

INTEGER*4 MLNGTH, LNGTH, MID, NID, MINTVS, ICNT

LOGICAL*1 ITEM, NEWID

DIMENSION ITEM (MLNGTH), NEWID (MID, MINTVS)

                   (Initialization Coding)

RETURN
```

The initialization coding may do any initialization the user wishes. However, before the RETURN is made, the identification array NEWID must be initialized to contain the identification for the first unit. The byte array NEWID contains NID bytes for each of the NINTVS levels, where NID and NINTVS are the numbers in Columns 31-35 and Columns 1-5, respectively, of the General Option Card. These are preset to blanks before SETITM is called. Thus the user must provide coding to place up to NID identification bytes into each of the NINTVS identification rows of NEWID. For example, if there are three levels (Overall, Category and Document) and the first document is numbered 4365 and is in Category ABC, then NEWID might be set up as follows:



The normal entry should have the following format:

```
     ENTRY  ITMID (*, *, *)
          (Item Identification Coding)

END
```

Normally, the user provides coding to identify an item in his input, to determine when the end of an input unit occurs and to determine when his input is complete. In the first case, the length of the item must be placed in LNGTH and the item itself into the first LNGTH bytes of the byte array ITEM. (In all cases, LNGTH must be less than MLNGTH.) Also, the frequency of this item may be placed in ICNT. At the beginning of the run, FRQNCY presets ICNT to one (1), which is the assumed normal case. Hence ICNT need only be set if the frequency differs from one. Note, however, that once ICNT has been changed, it is not automatically reset to one. Following this, the statement:

RETURN 1

must be coded. When the last item of one unit has been passed to FRQNCY and a new unit is to begin, then on the next entry to ITMID the identification array NEWID must be set up as above with the new unit's identification, and the statement

RETURN 2

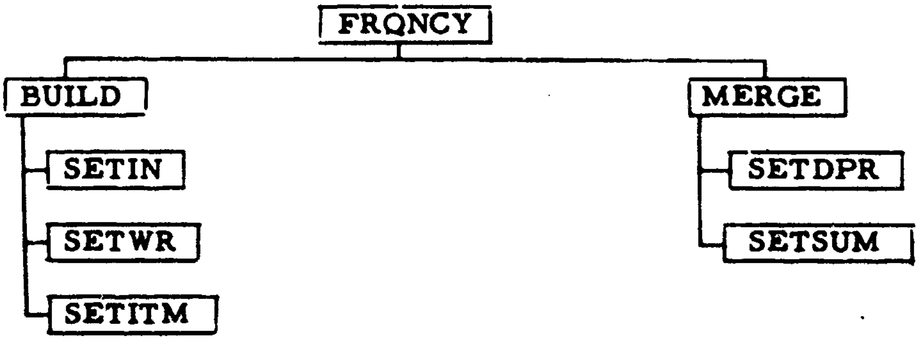
must be coded. Finally, when the last item of the last unit has been passed to FRQNCY and the input is complete, then on the next entry to ITMID any final processing the user may wish to perform must take place, and the statement

RETURN 3

must be coded. No further entries to ITMID will be made following this final exit.

7. PROGRAM ORGANIZATION

FRQNCY is organized so that it may be run in an overlay structure. A small setup and control program, FRQNCY, calls first the main subroutine BUILD for the sort phase and then the main subroutine MERGE for the merge phase. BUILD in turn calls three subroutines (including the user-written one, SETITM), while MERGE calls two. The overall organization is therefore of the following form:



A description of each of these routines and the functions they perform follows.

FRQNCY

FRQNCY is the setup and control routine for the entire program. It reads the General and Level Option Cards and performs a number of validity checks upon the information they contain. If any cards are incorrect, FRQNCY will abort the run. If the cards are all correct, then for a normal run, FRQNCY will call the BUILD subroutine to begin input processing. BUILD returns control after all input has been processed and placed in the merge data sets, and FRQNCY then calls MERGE to perform the merging and output functions. When MERGE has completed its operations, control is again returned to FRQNCY, which immediately executes a STOP statement to complete the run. In the case of a restart, FRQNCY bypasses BUILD completely and gives initial control to MERGE to continue the processing from the end of the last completed merge pass.

FRQNCY also sets up a number of control tables used in the run and places in COMMON many of the input parameters. The major tables generated are:

NAME: contains the name of each level.

IDENT: contains two subarrays for each level:

1. OLDID: identification of the last unit at this level.

2. NEWID: identification of the current unit at this level (set by SETITM).

SOUTS: contains four subarrays with information on the kinds of outputs desired at each level:

1. SUMMDS: number of data set for summary outputs or zero for no summary output.

2. GRINTV: token interval for growth rate distribution or zero for no growth rate distribution.

3. PRINDS: number of data set for detailed print outputs or zero for no detailed print output.

4. OUTPDS: number of data set for detailed outputs for further processing or zero for no detailed output for further processing.

DOUTS: contains seventeen subarrays with information on what is to be included in the outputs at each level. The first seven contain summary output data; the next five, detailed print output data; and the last five, data on detailed outputs for further processing.

BUILD

BUILD is the routine which accepts inputs from the user-written SETITM subroutine, places them in sort and accumulates the number of times each has appeared. BUILD begins processing by reading the Initial Dictionary Item Cards (if any were provided) and placing them into its internal dictionary with counts of zero. Any errors found in the sequence or number of cards input will cause an abort at this point. Following this, BUILD calls SETIN through its setup entry point (SETIN), to initialize the remainder of the dictionary, calls SETWR through its setup entry point (SETWR) to set internal switches controlling output formats, and finally calls SETITM through its setup entry point (SETITM) to allow initialization processing such as opening data sets, determination of the first unit's identification, etc. BUILD then completes its internal setup, positions the merge data sets to their beginning points and begins input processing.

Input processing is accomplished through interaction between BUILD and SETITM. BUILD calls SETITM through its processing entry (ITMID), and SETITM places the next input item in the array ITEM and optionally places its count (if any counts are unequal to one) in ICNT before making a normal return. BUILD then attempts to find this item in its dictionary. If successful, BUILD adds the contents of ICNT to the total count being accumulated for this item. If unsuccessful, then BUILD checks to make sure space exists for this item and adds it is possible. If not, then BUILD calls SETWR through its processing entry (WRDICT) to write out this partial dictionary and then calls SETIN through its processing entry (INDICT) to re-initialize the dictionary area for the next partial dictionary of this unit. Following thie, the item is added as the first item in the new partial dictionary.

If SETITM completes a unit, it places the next unit identification in the array NEWID and executes a RETURN 2 statement. BUILD then calls SETWR (through WRDICT) to write out this partial dictionary and then calls SETIN (through INDICT) to re-initialize the dictionary for the first partial dictionary of the next unit.

If SETITM completes the input, it executes a RETURN 3 statement. BUILD then calls SETWR (through WRDICT) to write out this final partial dictionary. It then closes the merge data sets, informs the operator that input processing is successfully completed, and returns control to FRQNCY.

SETITM uses a number of arrays to hold the items themselves, indexing information and counts. The major ones are:

BASE: contains the number of tokens counted so far in the present unit at each level.

OFLOW: contains counts for any items whose frequency is greater than MFREQ (set to 32767).

ITEM: contains the item currently being processed.

CWORKS: contains two subarrays with chain information:

1. CENTPT: contains an index value for the entry point in DWORKS of this chain of sequenced items.
2. CLNGTH: contains the current length of this DWORKS chain of sequenced items.

DWORKS: contains four subarrays with item information:

1. IFCHAR: contains an index value for the beginning byte of this item in DICT.
2. NENTX: contains an index value for the next DWORKS entry in this chain of sequenced items.
3. ILNGTH: contains the length (in bytes) of this item in DICT.
4. FREQ: contains the frequency within the current partial dictionary of this item in DICT.

FUSE: contains the count (in terms of total tokens) at which this item was first used in the present unit.

DICT: contains the items themselves.

An explanation of the method used in sequencing the input items is contained in Section 4.2.3 (pp. 33-38) of the document referenced in the Introduction. The technique is basically the directory-chain method with uni-directional chaining. The initial dictionary items are used to "prime" the chains with expected high-frequency words to reduce the average search time.

SETIN

The SETIN routine has two entry points, SETIN and INDICT. The SETIN entry is used only when BUILD is initializing the input processing phase and makes the addresses of dictionary arrays known within SETIN. The INDICT processing entry causes chain entry points and lengths in CWORKS to be reset and then causes the chaining indexes and frequency counts in DWORKS to be reset.

SETWR

SETWR also has two entry points, SETWR and WRDICT. The SETWR entry is used by BUILD during input processing phase initialization to set internal switches to correspond to the formats of the partial dictionaries to be written. The WRDICT entry is used whenever a partial dictionary must be written out to a merge data set. For each partial dictionary, a header containing the total frequency of all items within it, the array OLDID, and the array BASE, is written. Following this, SETWR follows the chains of sequenced dictionary items and, for each item, writes out its length and frequency, its first use (if selected), a one to indicate the number of next-lower-level uses (if selected) and the item itself. A dummy record with an item length of one, count(s) of zero and a single all-zero item byte is written to indicate the end of the sequenced string. Following completion of the write, the number of the merge data set is incremented so that the data sets are used in rotation.

MERGE

MERGE is the routine which merges the sequenced strings at a given level and calls necessary formatted output preparation routines as required. It consists essentially of four nested loops. The innermost loop merges sets of sequenced strings, the next loop controls the processing of each pass of the merge data sets, the next one controls the merging of all items at a given level and the outermost loop controls the levels from highest-numbered (detail) to lowest-numbered (overall). Regardless of the number of levels, processing is terminated when no more outputs remain to be produced; thus one can count units at any given level or levels of a hierarchy by merely specifying no outputs for the remaining levels.

MERGE begins processing by setting switches for the formats of the merge data sets. Then, entering the level loop, it sets summary and detail output switches for the highest-numbered level. The first input pass then begins, and partial dictionary strings are merged. Passes at this level continue until the maximum number of partial dictionary strings remaining to be merged in any unit at this level is less than or equal to the number of merge data sets. At this point, output switches are set so that the summary outputs and detail outputs specified are produced for these lowest-level units during the final merge pass at that level. This entire process then is repeated at the next higher level, with the completed dictionary strings for entire units at one level becoming the partial dictionary strings of the more comprehensive units at the next level. Outputs are produced on the final pass at each level until no more specified outputs remain, at which point MERGE returns control to FRQNCY.

Important arrays generated by MERGE are:

WORKS: contains six subarrays with information about the items being merged:

1. DSNO: contains number of the data set from which the item was read.

2. IFUSE: contains the count (in terms of total tokens) at which the item was first used in the present partial dictionary.

3. INUSE: contains the count of the number of next-lower-level units in which the item was used.

4. IFREQ: contains the frequency within the current partial dictionary of this item.

5. ILNGTH: contains the length (in bytes) of this item.

6. SEQ: contains the relative index of the order of this item among the items currently being merged.

IITEM: contains the actual items being merged.

OBASE: contains the new BASE array being generated for the merged string.

IBASE: contains the old BASE arrays for each string currently being merged.

SETDPR

SETDPR is the routine which generates and formats the detailed print outputs. It is first called through its SETDPR setup entry point during the level loop initialization of MERGE, and at this time it sets a switch to branch to the proper WRITE/FORMAT statement combinations for header and detail line writing. The DHEAD entry point is used when a new page must be started because a new unit is to be output, and this entry merely forces a heading to be printed by setting the line count to the maximum.

The normal item print entry is made on the last pass of each level through the DPRINT entry point. A check is made for line count overflow, and a header is written if necessary. Following this, the detail line is written and control returned to MERGE.

SETSUM

The SETSUM routine is used to accumulate and print summary outputs. At the beginning of each level, MERGE calls it through its SETSUM entry point to set its internal switches. At the beginning of processing for each unit on the last pass of each level, MERGE calls SETSUM through the SUMIN entry point to initialize all the summary data accumulation arrays to zero. As MERGE completes processing of each item of that unit, control is passed to SETSUM through the SUMCMP entry point to accumulate the specified summary statistics. Finally, upon completion of the unit, SETSUM is called through the SUMPRT entry point so that it may output the accumulated statistics.

Major arrays created by SETSUM are:

FCHS: contains two subarrays with first character frequency data:

1. TYPFCH: Type/first character data.
2. TOKFCH: Token/first character data.

ILGS: contains two subarrays with item length frequency data:

1. TYPILG: Type/item length data.
2. TOKILG: Token/item length data.

TYPTOK: contains type/token data.

TYPINT: contains growth rate data.

In addition to the routines and arrays described above, there are four named COMMON areas which contain important parameters used by a number of the routines. The labels of these areas, the manner in which they are used, and the constants they contain are as follows:

FRQALL (used throughout):

1. MMRGOR: Maximum merge order; presently set to 4.
2. MLNGTH: Maximum length (in bytes) of an input item; presently set to 255.
3. MINTVS: Maximum number of levels; presently set to 5.
4. MFREQ: Maximum capacity of the standard INTEGER*2 cell for accumulating frequencies; set to 32767 for System/360; frequencies greater than this become overflow items and are accumulated in INTEGER*4 cells of greater capacity.

5. MID: Maximum length of any level identification field; presently set to 16.

6. MRGOR: Merge order for this run; must not be greater than MMRGOR.

7. FIMRGS: Number of the lowest numbered input merge data set for this pass.

8. LIMRGS: Number of the highest-numbered input merge data set for this pass.

9. MRGS: Number of the current merge data set for this pass.

10. NINTVS: Number of levels for this run; must not be greater than MINTVS.

11. NID: Maximum length of any level identification field in this run; must not be greater than MID.

12. GPARS: Largest number of partial dictionaries in any unit at the present level.

13. NPARS: Number of partial dictionaries in the current unit at the present level.

14. TFREQ: Total frequency of all items in the current unit.

15. LLNGTH: Longest item at the present level.

16. LPERP: Number of lines to be written on pages of summary and detailed output.

17. COMMNT: Number of the data set to be used for comments and operator instructions.

18. FUSECT: "T" if first uses are to be counted at any level; "F" otherwise.

19. NUSECT: "T" if the number of next-lower-level uses is desired at any level; "F" otherwise.

FRQBLD (used by FRQNCY and BUILD):

1. MCHARS: Number of bytes available for item storage in the dictionary; presently set to 35000.

2. MDITMS: Number of different items which may be counted in one partial dictionary; presently set to 5000.

3. MCHNS: Number of chains of sequenced items which may be included in one partial dictionary (also one more than the number of initial dictionary items which may be supplied); presently set to 1000.

4. MCLNG: Maximum number of items in a chain before it will be split into two chains; should be specified as approximately (3*MDITMS)/(2*MCHNS).

5. MOITMS: Maximum number of items whose frequency is greater than MFREQ which may be counted in one partial dictionary; currently set to 100.

6. NINDIS: Number of initial dictionary items supplied in this run.

7. PARAMS: Number of the Parameter data set from which the input parameters will be read; presently set to 5.

8. CONCRD: Reserved for future use; should remain zero (0).

9. GONOGO: Internal indicator set during the process of checking input parameter cards.

10. SPITEM: Reserved for future use; should remain "F."

11. INEXAM: Reserved for future use; should remain "F."

FRQMRG (used by FRQNCY and MERGE):

1. MGRINT: Maximum number of growth rate intervals for which type counts can be accumulated (should be greater than the largest number obtained by dividing the number of tokens expected in units of each level by the growth rate interval specified on the Level Option Card for that level); presently set to 1000.

2. MTOKNS: Greatest token count for which the type/token distribution can be generated; presently set to 1000.

3. MSYMS: Number of unique first characters for which the two first character distributions can be generated; presently set to 256.

4. INTX: Internal counter used to indicate the current level number.

5. FOMRGS: Number of the lowest-numbered output merge data set for this pass.

6. LOMRGS: Number of the highest-numbered output merge data set for this pass.

BLD (used only within BUILD):

1. NCHARS: Number of bytes currently in use for item storage in the dictionary.

2. NINCHS: One more than the number of bytes in the initial dictionary items.

3. NDITMS: Current number of different items in the dictionary.

4. NCHNS: Current number of chains of sequenced items in the dictionary.

5. NOITMS: Current number of items whose frequency is greater than MFREQ.

8. PROGRAM ADAPTATION

There are several types of changes which may be required to adapt FRQNCY to run on a particular computer or to run for a particular purpose. This section is included to point out some of these potential changes and the ways in which FRQNCY may be modified to implement them.

Storage

One of the most likely ways in which a user may wish to modify FRQNCY is to adapt it to differing sizes of fast-access storage. Generally, efficiency can be increased by reducing the number of partial dictionaries required to count units at the lowest level. On the other hand, the size of the partial dictionaries which can be used is limited by the amount of fast-access storage available.

One of the two ways in which storage available for dictionaries can be adjusted is by making use of the organization of FRQNCY to run it as a planned overlay program. Section 7 described the program organization and indicated the manner in which the routines are used.

The second way in which storage available for dictionaries can be adjusted is by changing the sizes of major arrays to meet the requirements of the problem and/or the characteristics of the computer. To this end, three principles were used in writing the program:

1. All major arrays are defined and dimensioned at the lowest point possible consistent with the overlay structure.
2. Only the initial definitions of major arrays use absolute dimensions; all subsidiary definitions and subarrays use relative dimensions.

3. The parameters used to refer to the sizes of all major arrays are defined at the beginning of FRQNCY by Assignment statements and are referred to symbolically thereafter.

As a consequence of these principles, it is relatively simple to change the dimensions of major arrays, since at most assignment statements and a DIMENSION statement need be modified.

Each of the major arrays is discussed below, together with an indication of its dimensions, specified either as integers or symbolically. The integer dimensions shown below represent absolute dimensions which cannot be changed. The symbolic dimensions shown below indicate that the corresponding dimensions in the program may be changed. The symbols used in the dimension statements are identical to the symbols used in the Assignment statements at the beginning of the FRQNCY program. If the user wishes to change the dimension(s) of any array shown below with symbolic dimension(s), he must do two things. First, the Assignment statement(s) at the beginning of FRQNCY which assigns an integer value to the affected symbol(s) must be located, and they must be changed to assign the new value(s) desired. Second, the integer(s) actually used in the DIMENSION statement(s) defining the array(s) must be changed correspondingly. (It is important to note that in some cases a dimension change may affect more than one array and that in such cases all affected arrays must be changed.)

CWORKS (MCHNS, 2): This INTEGER*2 array is defined in BUILD and used to maintain information on the chains of sequenced items. It can be changed in size at the expense of changing the average chain length and thus the internal search time to find a particular item. The guide which should be used is that MCHNS should be approximately equal to (3*MDITMS)/(2*MCLNG).

DWORKS (MDITMS, 5): This INTEGER*2 array is defined in BUILD and used to maintain information on individual items. MDITMS is the maximum number of different items which can be counted in one partial dictionary and is hence a very critical parameter. The values of both MCHNS and MCHARS and hence the sizes of CWORKS and DICT (two of the largest arrays) also depend primarily on the value of MDITMS.

FUSE (MDITMS): This INTEGER*4 array is defined in BUILD and used to store the token count of an item's first use. It is not referred to unless first uses are being recorded for at least one level. Hence, if no first uses are being recorded, its dimension may be changed to 1 to save storage.

DICT (MCHARS): This LOGICAL*1 array is defined in BUILD and used to store the items themselves. The guide which should be used in selecting the value of MCHARS is that it should be the average item length expected (in terms of types, not tokens) multiplied by MDITMS.

TYPINT (MGRINT): This INTEGER*4 array is defined in SETSUM and used to store the growth rate information. MGRINT is the maximum number of growth rate intervals for which type counts can be accumulated and should be greater than the largest number obtained by dividing the number of tokens expected in units of each level by the growth rate interval specified on the Level Option Card for that level.

FCHS (MSYMS, 2): This INTEGER*4 array is defined in SETSUM and used to store the two first character distributions. MSYMS should be greater than or equal to the largest value occurring in the numeric representations of the first bytes of input items.

ILGS (MLNGTH, 2): This INTEGER*4 array is defined in SETSUM and used to store the two item length distributions. MLNGTH should be greater than or equal to the byte length of the longest input item.

Of the above arrays, the first four are defined in BUILD and the last four in SETSUM. Since the BUILD arrays are likely to occupy considerably more storage than the SETSUM ones, it is probably more productive to reduce their sizes if an overlay program structure is used as recommended above. If not, then attention should be paid to all arrays if storage is a problem.

Capability

A second reason for which FRQNCY may be modified is to change its capabilities. It may be desirable to handle more levels, to use a higher-order merge than allowable with the program as distributed, etc. While the arrays concerned here do not significantly affect storage requirements, they have been treated in the same manner as the arrays described above. Where they may be changed, both an Assignment statement at the beginning of FRQNCY and the integers actually used in the DIMENSION statements defining all affected arrays must be changed.

NAME (16, MINTVS): This LOGICAL*1 array is defined in FRQNCY and is used to store the name of each level. MINTVS represents the maximum number of levels.

IDENT (MID, MINTVS, 2): This LOGICAL*1 array is defined in FRQNCY and used to store the identifications supplied by SETITM for units at each level. MID represents the longest identification string allowable at any level.

SOUTS (MINTVS, 4): This INTEGER*4 array is defined in FRQNCY and used to store data set numbers and growth rate interval information.

DOUTS (MINTVS, 17): This LOGICAL*1 array is defined in FRQNCY and used to store output option information.

BASE (MINTVS): This INTEGER*4 array is defined in BUILD and used to store token count data from each level.

OFLOW (MOITMS): This INTEGER*4 array is defined in BUILD and used to store counts for items whose frequency is greater than MFREQ. MOITMS represents the maximum number of items whose frequency is greater than MFREQ which may be counted in one partial dictionary.

ITEM (MLNGTH): This LOGICAL*1 array is defined in BUILD and contains the current item. MLNGTH is the length of the longest item which can be counted; longer items are truncated to this length.

WORKS (MMRGOR, 10): This INTEGER*2 array is defined in MERGE and used to store information about the items being merged. MMRGOR is the maximum merge order.

IITEM (MLNGTH, MMRGOR): This LOGICAL*1 array is defined in MERGE and contains the items being merged.

OBASE (MINTVS): This INTEGER*4 array is defined in MERGE and contains the new BASE array being generated for the merged string.

IBASE (MMRGOR, MINTVS): This INTEGER*4 array is defined in MERGE and contains the old BASE arrays for each string currently being merged.

Of the above arrays, the first four are defined in FRQNCY, the next three in BUILD and the last four in MERGE.

Other Features

There are several miscellaneous parameters which the user may wish to modify for various reasons. These parameters are all included in the initial set of Assignment statements in FRQNCY but do not affect any array definitions:

MDS: This parameter is the highest data set number allowable. In general, it will be operating system dependent. It is currently set te 99 for compatibility with the OS/360 convention for Fortran data set naming.

PARAMS: This is the number of the data set from which the input parameters are read. It also is an operating system related parameter and is currently set to 5 for compatibility with the OS/360 cataloged procedures for Fortran runs.

MFREQ: This parameter is the maximum positive value which FRQNCY can store in the INTEGER*2 cells in which token counts are accumulated.

9. RESTRICTIONS

A few restrictions are applicable to use of the FRQNCY program. Each of the known restrictions is discussed below. It is important to note that, while FRQNCY is written entirely in Fortran IV, it has been tested only on the IBM System/360 Models 40 and 50. Different computers, operating systems or Fortran IV compilers may not produce the same results.

The most important restriction is that the Fortran IV compiler used to compile FRQNCY permit the LOGICAL*1 and INTEGER*2 type statements or reasonable equivalents. Extensive use is made of these to conserve storage, and, while FRQNCY could be modified to eliminate its dependence on these, it would use considerably more storage.

A second restriction concerning the compiler is that it must not produce coding to check array bounds at object time. Subarrays in general have been defined using EQUIVALENCE statements and minimum absolute dimensions. This was done to allow easy modification of array sizes to fit differing amounts of fast-access storage, since only the major array dimensions need be changed. If such check coding is produced, then all subarray dimensions must be explicitly stated in absolute form, and all must be changed whenever the corresponding major array dimensions are changed.

The third restriction on the compiler is that it permit the extended READ statement allowing an exit to a specified statement when an end-of-data-set condition is encountered.

The above three restrictions outline the major known deviations from compilers meeting USASI Fortran standards. One other minor incompatibility is that, in order to produce more readable output listings, both the vertical bar "|" and the underscore "_" have been used in FORMAT statements in the SETSUM and SETDPR subroutines. These can be changed to other characters or to blanks if necessary to prevent character set incompatibility.

10. EXAMPLE

This section is included to demonstrate setup and operational features of a typical run of FRQNCY under OS/360. It is divided into three sections, the first describing the setup peculiar to FRQNCY, the second describing the OS/360 setup, and the third describing the comments output during processing. The first and third would be applicable on any computer, while the second assumes a knowledge of OS/360 and would be applicable only to System/360 runs.

FRQNCY Setup

The run described was made to count words in individual documents occurring in a five-level hierarchial structure. It made use of a SETITM subroutine which was designed to identify words and pass them as the input items to FRQNCY. This SETITM subroutine placed five levels of input unit identification information into the NEWID array so that FRQNCY could theoretically have performed counts at all levels of the hierarchy from individual documents to the overall input. However, in this particular run only document counts were desired, so that outputs for only that level (level 5) were requested.

The first card prepared was the General Option Card. Five levels were present in the input structure, so columns 1-5 of this card were "ƀƀƀƀ5."* No initial dictionary items were supplied, so columns 6-10 were "ƀƀƀƀ0". The maximum length chain was 10 (eight would have been a slightly better choice), so columns 11-15 were "ƀƀƀ10". A two-way merge was used, so columns 16-20 were "ƀƀƀƀ2". The first merge data set was 9 (and hence the last was 12), so columns 21-25 were "ƀƀƀƀ9". No more than fifty lines of output were to be placed on a page, so columns 26-30 were "bbb50". The longest identification field at any of the five levels was 6, so columns 31-35 were "ƀƀƀƀ6". The data set for comments was to be the on-line printer, which in the standard OS/360 Fortran procedure is data set 6, so columns 36-40 were "ƀƀƀƀ6". Finally, columns 41-45 were "ƀƀƀƀ0", column 47 was "F" and column 49 was "F" to deselect options not presently included.

The next five cards in the FRQNCY input were the five Level Option Cards. They were in order from the highest level (Level 1) to the lowest (level 5). For the highest level, the name was "EVALOVAL", so columns 1-16 of the first card were "ƀƀƀƀEVALOVALƀƀƀƀ". No outputs were desired at this level, so columns 19, 36, 39 and 47 were punched with a zero. The second, third, and fourth cards were punched identically except for the name fields in columns 1-16. These were, respectively, punched as

* Note: The representation "ƀ" is used to indicate the character "blank".

"ƀƀƀƀEVAL1STLƀƀƀƀ", "ƀƀƀƀEVAL2NDLƀƀƀƀ" and ƀƀƀƀEVAL3RDLƀƀƀƀ".

The fifth card contained additional information, since outputs were desired at this level. The name was "EVALDOCT", so columns 1-16 were punched "ƀƀƀƀEVALDOCTƀƀƀƀ". No summary outputs were desired, so column 19 was punched with a zero. Column 36 was also punched with a zero, since no growth rate distribution was needed. Since no print output was wanted, column 39 was also punched with a zero. To produce a tape for further processing as data set number 21, columns 45-47 were punched "ƀ21". Only the item length and its frequency were needed, so columns 48-52 were punched "TTFFF".

Since no Initial Dictionary Item Cards were to be included, these six cards comprised the entire set of input parameters. They were placed in the input stream as OS/360 SYSIN input and hence became data set 5 according to the standard OS/360 Fortran cataloged procedure.

OS/360 Setup

Figure 1 shows the OS/360 Job Control Language cards which were provided to compile, link edit and run the FRQNCY job described above. The reader is referred to the IBM manual, Job Control Language (IBM Form C28-6539) for a fuller explanation of

```
//J5049F          JOB    WC1,BAKER,MSGLEVEL=1
//                EXEC   PROC=FORTGCLG,
//                 PARM.FORT='NAME=FRQNCY,MAP,DECK',                        X
//                 PARM.LKED='MAP,LIST'                                     X
//FORT.SYSIN      DD     *
                 FORTRAN
                 Source Deck
/*
//LKED.SYSIN      DD     *
                 LIBRARY    *(INSPIT,ININEX,INCONC,INEXM,SPITM,CONCOR)
/*
//GO.FT08F001 DD      DSNAME=DOCSOB,
//                DISP=(OLD,KEEP),                                          X
//                UNIT=183,                                                 X
//                VOLUME=SER=005349                                         X
//GO.FT09F001 DD      DSNAME=MERG01,
//                UNIT=SYSDA,                                               X
//                VOLUME=SER=SYSR01,                                        X
//                DISP=(NEW,DELETE),                                        X
//                SPACE=(CYL,(20,10)),                                      X
//                DCB=(,RECFM=FB,LRECL=26,BLKSIZE=3614,BUFNO=2)             X
//GO.FT10F001 DD      DSNAME=MERG02,
//                UNIT=SYSDA,                                               X
//                VOLUME=SER=SYSR03,                                        X
//                DISP=(NEW,DELETE),                                        X
//                SPACE=(CYL,(20,10)),                                      X
//                DCB=(*.FT09F001)                                          X
//GO.FT11F001 DD      DSNAME=MERG03,
//                UNIT=180,                                                 X
//                VOLUME=SER=005260,                                        X
//                DISP=(NEW,KEEP),                                          X
//                DCB=(*.FT09F001)                                          X
//GO.FT12F001 DD      DSNAME=MERG04,
//                UNIT=181,                                                 X
//                VOLUME=SER=005095.                                        X
//                DISP=(NEW,KEEP),                                          X
//                DCB=(*.FT09F001)                                          X
//GO.FT21F001 DD      DSNAME=EVALDOCT,
//                UNIT=182,                                                 X
//                VOLUME=SER=005250,                                        X
//                DISP=(NEW,DELETE),                                        X
//                LABEL=(1,SL),                                             X
//                DCB=(,RECFM=FB,LRECL=26,BLKSIZE=3614,BUFNO=2)             X
//GO.SYSIN        DD     *
                 Parameter
                 Data Set
/*
//
```

Figure 1. OS/360 Input

this input, which is presented here primarily to suggest one way in which this job could be set up.

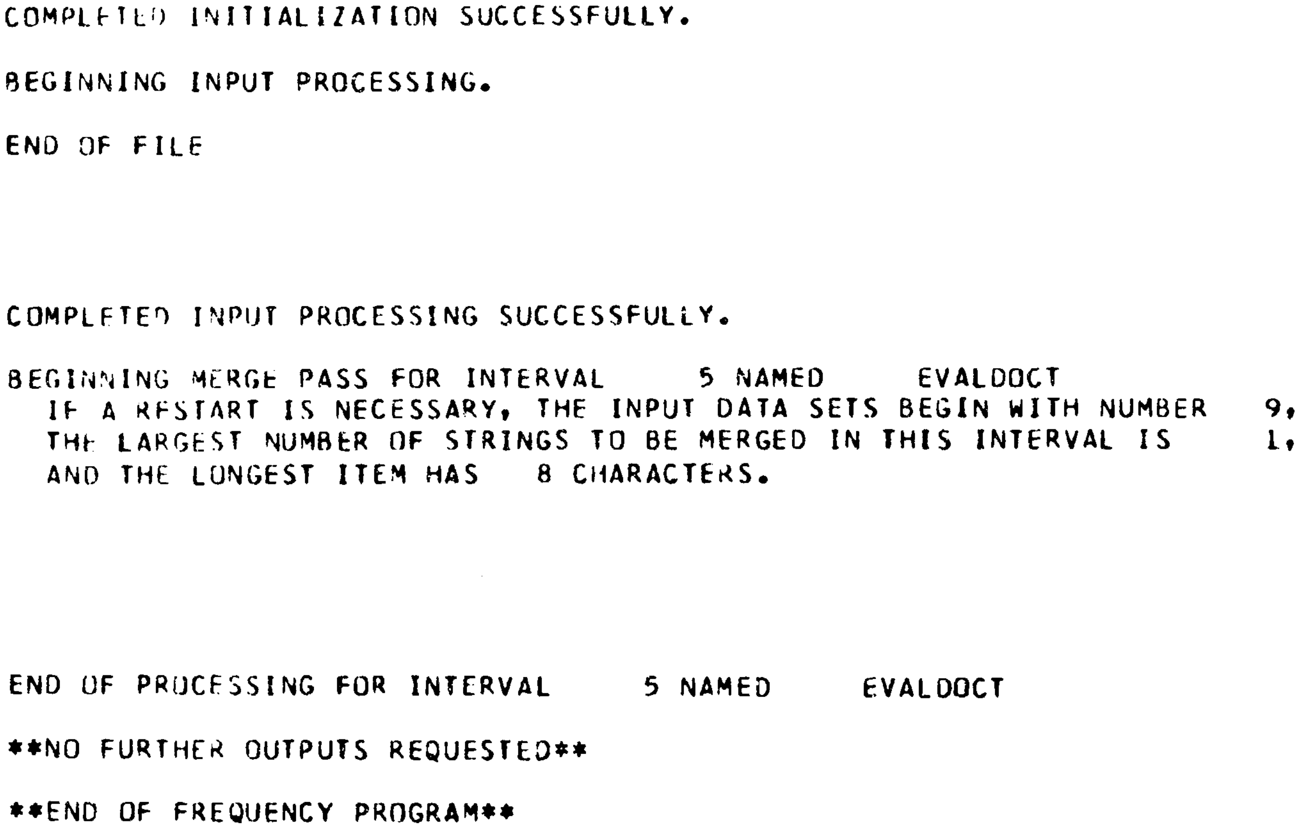
FRQNCY Comments

Figure 2 shows the comments which FRQNCY provides at important points in the processing. These comprise the normal comments expected in a complete, correct run. Error comments provided, as part of the input diagnosis process or at the termination of a run where errors occurred, are generally self-explanatory and are not included here.

After signing on in the first comment, FRQNCY prints out the options specified on the General Option Card. This is followed by listing of all options specifked on each of the Level Option Cards. If no errors have been found (each error is flagged with asterisks and indicated on the line containing the option listing), then FRQNCY prints the comments on successful initialization and beginning of processing. The comment "END OF FILE" which follows was printed by the SETITM subroutine and not by FRQNCY. However, following that comment, which was printed when SETITM encountered the end of the input data set, FRQNCY printed the comment on completion of input processing.

```
BEGINNING FREQUENCY PROGRAM.

  GENERAL OPTIONS ARE:
        5 INTERVALS.
        0 INITIAL DICTIONARY ITEMS.
       10 ITEMS IN MAXIMUM LENGTH CHAIN.
        2 IS MERGE ORDER.
        9 IS FIRST MERGE DATA SET.
       12 IS LAST MERGE DATA SET.
        6 IS COMMENTS DATA SET.
       50 LINES PER PAGE ON PRINTER OUTPUT.
        6 IDENTIFICATION CHARACTERS AT EACH INTERVAL.
          NO CONCORDANCE DATA SET.
          NO SPECIAL ITEM CHECK.
          NO INPUT ITEM EXAMINATION.



  OPTIONS LISTED BELOW WERE SELECTED FOR INTERVAL      1 NAMED      EVALUVAL


  OPTIONS LISTED BELOW WERE SELECTED FOR INTERVAL      2 NAMED      EVAL1STL


  OPTIONS LISTED BELOW WERE SELECTED FOR INTERVAL      3 NAMED      EVAL2NDL


  OPTIONS LISTED BELOW WERE SELECTED FOR INTERVAL      4 NAMED      EVAL3RDL


  OPTIONS LISTED BELOW WERE SELECTED FOR INTERVAL      5 NAMED      EVAL[illegible]

       21 IS OUTPUT DATA SET WITH EACH ITEM AND ITS:
              LENGTH
              RAW FREQUENCY
```

Figure 2. FRQNCY Comments



```
COMPLETED INITIALIZATION SUCCESSFULLY.

BEGINNING INPUT PROCESSING.

END OF FILE



COMPLETED INPUT PROCESSING SUCCESSFULLY.

BEGINNING MERGE PASS FOR INTERVAL      5 NAMED      EVALDOCT
   IF A RESTART IS NECESSARY, THE INPUT DATA SETS BEGIN WITH NUMBER    9,
   THE LARGEST NUMBER OF STRINGS TO BE MERGED IN THIS INTERVAL IS      1,
   AND THE LONGEST ITEM HAS   8 CHARACTERS.



END OF PROCESSING FOR INTERVAL      5 NAMED      EVALDOCT

**NO FURTHER OUTPUTS REQUESTED**

**END OF FREQUENCY PROGRAM**
```

Figure 2. FRQNCY Comments (Continued)

At the beginning of each merge pass, FRQNCY prints the comment presenting the restart information which must be used if the pass is aborted. At the completion of the merge for each level, the comment on end of processing is printed. Finally, if the last level has been processed, the end of program comment is printed. In this case, since termination was based on the fact that no further outputs were requested at higher levels, a comment to that effect precedes the end comment.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Federal Systems Division International Business Machines Corporation Gaithersburg, Md. 20760	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

RESEARCH ON AUTOMATIC CLASSIFICATION, INDEXING AND EXTRACTING

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Annual Progress Report

5. AUTHOR(S) *(Last name, first name, initial)*

Baker, F. T., Williams, John H., Jr.

6. REPORT DATE August 1968	7a. TOTAL NO. OF PAGES 60	7b. NO. OF REFS None

8a. CONTRACT OR GRANT NO. NONR 4456(00) b. PROJECT NO. c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES

Qualified requesters may obtain copies of this report from DDC. Other qualified users shall request copies of this report from the originator.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY Information Systems Branch Office of Naval Research Dept. of the Navy, Washington, D. C.

13. ABSTRACT

To support studies in automatic indexing, classification and extracting, a general purpose frequency program has been developed to further our theoretical and practical understanding of text word distributions. While the program is primarily designed for counting strings of character-oriented data, it can be used without change for counting any items which can be represented in an integral number of characters. Counts may be obtained simultaneously at several levels of detail, such as for sentences, paragraphs, chapters and entire documents. Both printed outputs and outputs for further computer processing may be obtained, and a variety of summary and detailed outputs are available.

The program, titled FRQNCY, is written in the Fortran IV language and has been compiled and run on the IBM System/360 using Fortran IV (G) and the System/360 Operating System. It uses at least one feature of the IBM System/360 Fortran IV language (LOGICAL*1 and INTEGER*2 variables) which is not in USASI Fortran and hence may not compile or run under other Fortran systems. The program is extensively parametrized to allow its efficient use on computers with varying amounts of immediate-access storage and input/output equipment.

This report is a complete writeup of the frequency program. It covers the purpose and usage of the program and also describes its organization and internal operation. Finally, guidelines for modifying the program or adapting it to different computers are also included.

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Information Retrieval						
Information Systems						
Documentation						
Libraries						
Word Association						
Correlation Techniques						
Dictionaries						
Vocabulary						
Information Sciences						
Programming (computers)						
Word Frequency						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
_______________________________."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
_______________________________."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
_______________________________."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.